University of Macedonia
Economic and Social Studies
Master in Information Systems

Networking Technologies

Professors: A.A. Economides
A. Pomportsis

<u>SUBJECT :</u>

WEB MULTIMEDIA DELIVERY PROTOCOLS

<u>STUDENT</u>

Tsougas Asterios 25/04

THESSALONIKI 2005

Πανεπιστήμιο Μακεδονίας
Οικονομικών και Κοινωνικών Επιστημών
Μεταπτυχιακό Πρόγραμμα στα Πληροφορικά Συστήματα

ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΚΤΥΩΝ

Καθηγητές : Α.Α. Οικονομίδης
Α. Πομπότσης

<u>ΘΕΜΑ :</u>

WEB MULTIMEDIA DELIVERY PROTOCOLS

<u>ΦΟΙΤΗΤΗΣ:</u>

Τσούγκας Αστέριος 25/04

ΘΕΣΣΑΛΟΝΙΚΗ 2005

CONTENTS

Περίληψη

Είναι γεγονός ότι η ύπαρξη του διαδικτύου Internet από το 1969, όπου δημιουργήθηκε μέχρι τις αρχές του 1990 ήταν άγνωστη στο ευρύ κοινό. Με την έλευση του web και των ιστοσελίδων και με την μετέπειτα εξέλιξη του παγκόσμιου ιστού από μια απλή παράθεση κειμένου σε ένα ζωντανό πολυμεσικό περιβάλλον εμφανίστηκαν νέα πρωτόκολλα για να στηρίξουν τις εφαρμογές ήχου, εικόνας, ζωντανής μετάδοσης μέσω του internet. Είναι τα λεγόμενα Web multimedia internet protocols. Αυτά είναι βασικά τα RSVP, RTCP, RTP, RTSP, SAP, SDP και SIP. Λόγω της έλλειψης ελληνικής βιβλιογραφίας και των ιδιαίτερα δύσκολων επεξηγήσεων πολλών επιστημονικών ορισμών θεωρώ πως η ανάλυση θα ήταν πιο σωστό να παραμείνει ως έχει στην αγγλική για να μην βγει κάτι που δεν θα έχει νόημα

Abstract

It is fact that the existence of Internet from 1969, where it was created up to the beginnings 1990 was unknown in the wide public. With the arrival of web with its web pages and with the later development of world web from a simple apposition of text in a live multimedia environment were presented new protocols in order to they support the applications of sound, picture, live transmission via internet. These protocols called Web multimedia internet protocols. These are basically the RSVP, RTCP, RTP, RTSP, SAP, SDP and SIP. Because the lack of Greek bibliography and particularly difficult explanations of many scientific definitions I consider that the analysis it would be righter to remain as such in English in order to it does not come out something that will not have meaning.

Multimedia Networking: Goals and Challenges

Computer networks were designed to connect computers on different locations so that they can share data and communicate. In the old days, most of the data carried on networks was textual data. Today, with the rise of multimedia and network technologies, multimedia has become an indispensable feature on the Internet. Animation, voice and video clips become more and more popular on the Internet. Multimedia networking products like Internet telephony, Internet TV, video conferencing have appeared on the market. In the future, people would enjoy other multimedia products in distance learning, distributed simulation, distributed work groups and other areas.

For net workers, multimedia networking is to build the hardware and software infrastructure and application tools to support multimedia transport on networks so that users can communicate in multimedia. Multimedia networking will greatly boost the use the of computer as a communication tool. We believe someday multimedia networks will replace telephone, television and other inventions that had dramatically changed our life.

The real-time challenge

However, multimedia networking is not a trivial task. We can expect at least three difficulties. First, compared with traditional textual applications, multimedia applications usually require much higher bandwidth. A typical piece of 25 second 320x240 QuickTime movie could take 2.3MB, which is equivalent to about 1000 screens of textual data. This is unimaginable in the old days when only textual data is transmitted on the net.

Second, most multimedia applications require the real-time traffic. Audio and video data must be played back continuously at the rate they are sampled. If the data does not arrive in time, the playing back process will stop and human ears and eyes can easily pick up the artifact. In Internet telephony, human beings can tolerate a latency of about 250 millisecond. If the latency exceeds this limit, the voice will sound like a call routed over a long satellite circuit and users will complain about the quality of the call. In addition to the delay, network congestion also has more serious effects on real-time traffic. If the network is congested, the only effect on non-realtime traffic is that the transfer takes longer to complete, but real-time data becomes obsolete and will be dropped if it doesn't arrive in time. If no proper reaction is not taken, the retransmission of lost packets would aggravate the situation and jam the network.

Third, multimedia data stream is usually bursty. Just increasing the bandwidth will not solve the burstiness problem. For most multimedia applications, the receiver has a limited buffer. If no measure is taken to smooth the data stream, it may overflow or underflow the application buffer. When data arrives too fast, the buffer will overflow and the some data packets will be lost, resulting in poor quality. When data arrives too slow, the buffer will underflow and the application will starve.

Contrary to the high bandwidth, real-time and bursty traffic of multimedia data, in real life, networks are shared by thousands and millions of users, and have limited bandwidth, unpredictable delay and availability. How to solve these conflicts is a challenge multimedia networking must face. The possibility of answering this challenge comes from the existing network software architecture and fast developing hardware. The basis of Internet, TCP/IP and UDP/IP, provides a range of services that multimedia applications can use. Fast networks like Gigabit Ethernet, FDDI, and ATM provide high bandwidth required by digital audio and video. So the design of real-time protocols for multimedia networking becomes imperative before the multimedia age comes.

Multimedia over Internet

There are other ways to transmit multimedia data, like dedicated links, cables and ATM. However, the idea of running multimedia over Internet is extremely attractive.

Dedicated links and cables are not practical because they require special installation and new software. Without an existing technology like LAN, WAN, the software development will be extremely expensive. ATM was said to be the ultimate solution for multimedia because it supports very high bandwidth, is connection-oriented and can tailor different level of quality of service to different type of applications. But at this moment, very few users have ATM networks reaching their organization, even fewer have ATM connections to their desktops.

On the other hand, the Internet is growing exponentially. The well established LAN and WAN technologies based on IP protocol suite connect bigger and bigger networks all over the world to the Internet. In fact, Internet has become the platform of most networking activities. This is the primary reason to develop multimedia protocols over Internet. Another benefit of running multimedia over IP is that users can have integrated data and multimedia service over one single network, without investing on another network hardware and building the interface between two networks.

At current time, IP and Ethernet seem to be more favoured in the desktops and LANs, with ATM in wide area networks. As a shared datagram network, Internet is not naturally suitable for real-time traffic. To run multimedia over Internet, several issues must be solved. First, multimedia means extremely dense data and heavy traffic. The hardware has to provide enough bandwidth.

Second, multimedia applications are usually related to multicast, i.e., the same data stream, not multiple copies, is sent a group of receivers. For example, in video conference, the video data need to be sent to all participants at the same time. Live video can be sent to thousands of recipients. The protocols designed for multimedia applications must take into account multicast in order to reduce the traffic.

Third, the price tag attached shared network resources is unpredictable availability. But real-time applications require guaranteed bandwidth when the transmission takes place. So there must be some mechanisms for real-time applications to reserve resources along the transmission path.

Fourth, Internet is a packet-switching datagram network where packets are routed independently across shared networks. The current technologies cannot guarantee that real-time data will reach the destination without being jumbled and jerky. Some new transport protocols must be used to take care of the timing issues so that audio and video data can be played back continuously with correct timing and synchronization.

Fifth, there should be some standard operations for applications to manage the delivery and present the multimedia data. The answers to the above issues are the protocols that will be discussed in this paper.

The solution

The Internet carries all types of traffic, each type has different characteristics and requirements. For example, a file transfer application requires that some quantity of data is transferred in an acceptable amount of time, while Internet telephony requires that most packets get to the receiver in less than 0.3 seconds. If enough bandwidth is available, best-effort service fulfils all of these requirements. When resources are scarce, however, real-time traffic will suffer from the congestion.

The solution for multimedia over IP is to classify all traffic, allocate priority for different applications and make reservations. The Integrated Services working group in the IETF (Internet Engineering Task Force) developed an enhanced Internet service model called Integrated Services that includes best-effort service and real-time service, see RFC 1633. The real-time service will enable IP networks to provide quality of service to multimedia applications. Resource ReServation Protocol (RSVP), together with Real-time Transport Protocol (RTP), Real-Time Control Protocol (RTCP), Real-Time Streaming Protocol (RTSP), provides a working foundation for real-time services. Integrated Services allows applications to configure and manage a single infrastructure for multimedia applications and traditional applications. It is a comprehensive approach to provide applications with the type of service they need and in the quality they choose.

This paper, which takes many materials from corresponding Internet Drafts and RFCs, is a detailed review of the seven protocols (RTP, RTCP, RTSP, RSVP, SAP, SDP and SIP).

RSVP --- Resource ReSerVation Protocol

RSVP is the network control protocol that allows data receiver to request a special end-to-end quality of service for its data flows. Real-time applications use RSVP to reserve necessary resources at routers along the transmission paths so that the requested bandwidth can be available when the transmission actually takes place. RSVP is a main component of the future Integrated Services Internet which can provide both best-effort and real-time service[ISInt93].

Development

RSVP design has been a joint effort of Xerox Corp.'s Palo Alto Research Center (PARC), MIT, and Information Sciences Institute of University of California (ISI). The RSVP specification was submitted to the Internet Engineering Steering Group (IESG) for consideration as a Proposed RFC in November 1994. In September 1997, RSVP Version 1 Functional Specification and several other related internet drafts were approved as Proposed Standards They are:

•RFC 2205, Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification l

•RFC 2206, RSVP Management Information Base using SMIv2 (RFC 2206) l

•RFC 2207, RSVP Extensions for IPSEC Data Flows l

•RFC 2208, RSVP Version 1 Applicability Statement Some Guidelines on Deployment l

•RFC 2209, RSVP Version 1 Message Processing Rules l

The RSVP working group of the IETF is developing other protocols to be used with RSVP.

How does RSVP work?

RSVP is used to set up reservations for network resources. When an application in a host (the data stream receiver) requests a specific quality of service (QoS) for its data stream, it uses RSVP to deliver its request to routers along the data stream paths. RSVP is responsible for the negotiation of connection parameters with these routers. If the reservation is setup, RSVP is also responsible for maintaining router and host states to provide the requested service.

Each node capable of resource reservation has several local procedures for reservation setup and enforcement (see Figure 1). *Policy control* determines whether the user has administrative permission to make the reservation. In the future, authentication, access control and accounting for reservation will also be implemented by policy control. *Admission control* keeps track of the system resources and determines whether the node has sufficient resources to supply the requested QoS.

The RSVP daemon checks with both procedures. If either check fails, the RSVP program returns an error notification to the application that originated the request. If both checks succeed, the RSVP daemon sets parameters in the packet classifier and packet scheduler to obtain the requested QoS. The *packet classifier* determines the QoS class for each packet and the *packet scheduler* orders packet transmission to achieve the promised QoS for each stream.

RSVP daemon also communicate with the routing process to determine the path to send its reservation requests and to handle changing memberships and routes.

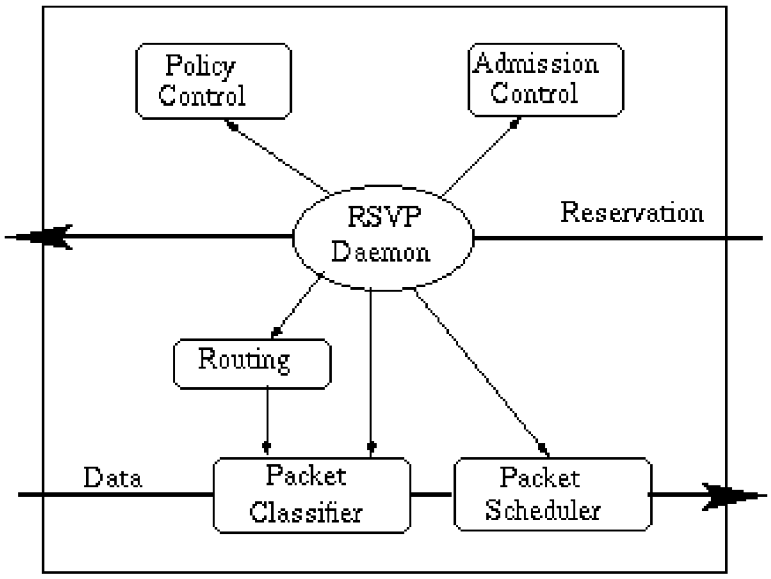


Figure 1: reservation at a node on the data flow path

This reservation procedure is repeated at routers along the reverse data stream path until the reservation merges with another reservation for the same source stream.

Reservations are implemented through two types of RSVP messages: PATH and RESV. The PATH messages are sent periodically from the sender to the multicast address. A PATH message contains *flow spec* to describe sender template (data format, source address, source port) and traffic characteristics. This information is used by receivers to find the reverse path to the sender and to determine what resources should be reserved. Receivers must join the multicast group in order to receive PATH messages.

RESV messages are generated by the receivers and contains reservation parameters including *flow spec* and *filter spec.* The filter spec defines what packets in the flow should be used by the packet classifier. The flow spec is used in packet scheduler and its content depends on the service. RESV messages follow the exact reverse path of PATH messages, setting up reservations for one or more senders at every node.

The reservation states RSVP builds at the routers are *soft states.* The RSVP daemon needs to send refresh messages periodically to maintain the reservation states. The absence of refresh message within a certain time will destroy the reservation state. By using soft states, RSVP can easily handle changing memberships and routes.

The reservation requests are initiated by the receivers. They do not need to travel all the way to the source of the sender. In stead, it travels upstream until it meets another reservation request for the same source stream, then merges with that reservation. Figure 2 shows how the reservation requests merge as they progress up the multicast tree.

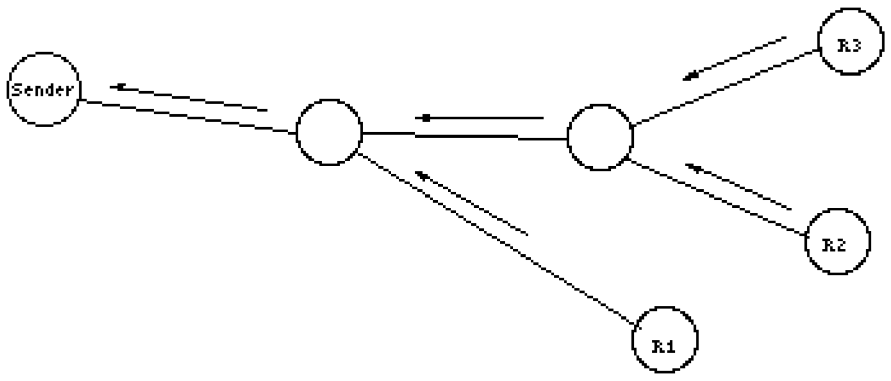


Figure 2: reservation merging.

This reservation merging leads to the primary advantage of RSVP: scalability---a large number of users can be added to a multicast group without increasing the data traffic significantly. So RSVP can scale to large multicast groups and the average protocol overhead decreases as the number of participantsincreases.

The reservation process does not actually transmit the data and provide the requested quality of service. But through reservation, RSVP guarantees the network resources are available when the transmission actually takes place.

Although RSVP sits on top of IP in the protocol stack, it is not a routing protocol, but rather an internet control protocol. Actually, RSVP relies on the underlying routing protocols to find where it should deliver the reservation requests. RSVP is also intended to cooperate with unicast and multicast routing protocols. When the RSVP-managed flow changes its path, the routing module will notify the RSVP module of the route changes. Therefore, RSVP can quickly adjust the resource reservation to new routes.

The delivery of reservation parameters is different from the determination of these parameters. How to set the connection parameters to achieve the requested QoS is the task of QoS control devices, the role of RSVP is just a general facility to distribute these parameters. Since different applications may have different QoS control devices, RSVP is designed to treat these QoS parameters as opaque data to be delivered to and interpreted by the control modules at the routers. This logical separation of QoS control devices and distribution facility simplifies the design of RSVP and makes it more adaptive to new network technologies and applications.

RSVP Features

•RSVP flows are simplex.

RSVP distinguishes senders and receivers. Although in many cases, a host can act both as a sender and as a receiver, one RSVP reservation only reserves resources for
data streams in one direction.

•RSVP supports both multicast and unicast, and adapts to changing memberships and routes.

RSVP is designed for both multicast and unicast. Since the reservations are initiated by the receivers and the reservation states are soft, RSVP can easily handle changing memberships and routes. A host can send IGMP (Internet Group Management Protocol) messages to join a multicast group. Reservation merging enabkes RSVP to scale to large multicast groups without causing heavy overhead for the sender.

•RSVP is receiver-oriented and handles heterogeneous receivers.

In heterogeneous multicast groups, receivers have different capacities and levels of QoS. The receiver oriented RSVP reservation requests facilitate the handling of heterogeous multicast groups. Receivers are responsible for choosing its own level of QoS, initiating the reservation and keeping it active as long as it wants. The senders divide traffic in several flows, each is a separate RSVP flow with different level of QoS. Each RSVP flow is homogeneous and receivers can choose to join one or more flows. This approach makes it possible for heterogeneous receivers to request different QoS tailored to their particular capacities and requirements.

•RSVP has good compatibility.

Efforts have been made to run RSVP over both IPv4 and IPv6. It provides opaque transport of traffic control and policy control messages in order to be more adaptive to new technologies. It also provides transparent operation through non-supporting regions.

Further discussion on the objectives and general justification for RSVP design are presented in [RSVP93 ,ISInt93].

RSVP Interfaces

RSVP communicates with both applications at the end hosts and various components inside network routers. For application programmers, the most important part is the client application programming interface. In an implementation developed by the ISI and Sun Microsystems, the following fundamental functions make up an RSVP client library called RAPI (RSVP Application Programming Interface):

•`rapi_session()` creates and initiates an API session and return a session handle for further reference.

•`rapi_sender()` is called by a sender application to specifies the parameters of its data flow. The RSVP daemon at the sender will use this flow spec to create PATH messages for this flow.

•`rapi_reserve()` is used to make, modify, or delete a reservation.

•`rapi_release()` asks the RSVP daemon to tear down the reservation.

RTP --- Real-time Transport Protocol

Realtime transport protocol (RTP) is an IP-based protocol providing support for the transport of real-time data such as video and audio streams. The services provided by RTP include time reconstruction, loss detection, security and content identification. RTP is primarily designed for multicast of real-time data, but it can be also used in unicast. It can be used for one-way transport such as video-on-demand as well as interactive services such as Internet telephony.

RTP is designed to work in conjunction with the auxiliary control protocol RTCP to get feedback on quality of data transmission and information about participants in the on-going session.

Development

Attempts to send voice over networks began in early 70's. Several patents on packet transmission of speech, time stamp and sequence numbering were granted in 70's and 80's. In 1991, a series of voice experiments were completed on DARTnet. In August 1991, the Network Research Group of Lawrence Berkeley National Laboratory released an audio conference tool vat for DARTnet use. The protocol used was referred later as *RTP version 0*.

In December 1992 Henning Schulzrinne, GMD Berlin, published *RTP version 1*. It went through several states of Internet Drafts and was finally approved as an Proposed Standard on November 22, 1995 by the IESG. This version was called *RTP version 2* and was published as:

•**RFC 1889**, *RTP: A Transport Protocol for Real-Time Applications*

•**RFC 1890**, *RTP Profile for Audio and Video Conferences with Minimal Control*

On January 31, 1996, Netscape announced "Netscape LiveMedia" based on RTP and other standards. Microsoft claims that their NetMeeting Conferencing Software supports RTP. The latest extensions have been made by an industry alliance around Netscape Inc., who uses RTP as the basis of the Real Time Streaming Protocol RTSP.

How does RTP work

As discussed in the first section, Internt is a shared datagram network. Packets sent on the Internet have unpredictable delay and jitter. But multimedia applications require appropriate timing in data transmission and playing back. RTP provides timestamping, sequence numbering, and other mechanisms to take care of the timing issues. Through these mechanisms, RTP provides end-to-end transport for real-time data over datagram network.

*Timestamping i*s the most important information for real-time applications. The sender sets the timestamp according to the instant the first octet in the packet was sampled. Timestamps increase by the time covered by a packet. After receiving the data packets, the receiver uses the timestamp to reconstruct the original timing in order to play out the data in correct rate. Timestamp is also used to synchronize different streams with timing properties, such as audio and video data in MPEG. However, RTP itself is not responsible for the synchronization. This has to be done in the application level. UDP does not deliver packets in timely order, so *sequence numbers* are used to place the incoming data packets in the correct order. They are also used for packet loss detection. Notice that in some video format, when a video frame is split into several RTP packets, all of them can have the same timestamp. So just timestamp is not enough to put the packets in order.

The *payload type identifier* specifies the payload format as well as the encoding/compression schemes. From this payload type identifier, the receiving application knows how to interpret and play out the payload data. Default payload types are defined in RFC 1890. Example specifications include PCM, MPEG1/MPEG2 audio and video, JPEG video, Sun CellB video, H.261 video streams, et al. More payload types can be added by providing a profile and payload format specification. At any given time of transmission, an RTP sender can only send one type of payload, although the payload type may change during transmission, for example, to adjust to network congestion.

Another function is *source identification.* It allows the receiving application to know where the data is coming from. For example, in an audio conference, from the source identifier a user could tell who is talking.

The above mechanisms are implemented through the RTP header. Figure 3 shows a RTP packet encapsulated in a UDP/IP packet.

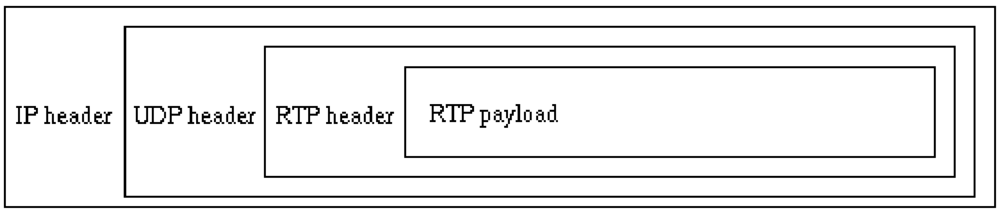


Figure 3: RTP data in an IP packet

RTP is typically run on top of UDP to make use of its multiplexing and checksum functions. TCP and UDP are two most commonly used transport protocols on the Internet. TCP provides a connection-oriented and reliable flow between two hosts, while UDP provides a connectionless but unreliable datagram service over the network. UDP was chosen as the target transport protocol for RTP because of two reasons. First, RTP is primarily designed for multicast, the connection-oriented TCP does not scale well and therefore is not suitable. Second, for real-time data, reliability is not as important as timely delivery. Even more, reliable transmission provided by retransmission as in TCP is not desirable. For example, in network congestion, some packets might get lost and the application would result in lower but acceptable quality. If the protocol insists a reliable transmission, the retransmitted packets could possibly increase the delay, jam the network, and eventually starve the receiving application.

RTP and RTCP packets are usually transmitted using UDP/IP service. However, efforts have been made to make them transport-independent so they can be also run on CLNP(Connectionless Network Protocol), IPX (Internetwork Packet Exchange), AAL5/ATM or other protocols.

In practice, RTP is usually implemented within the application. Many issues, such as lost packet recovery and congestion control, have to be implemented in the application level.

To set up an RTP session, the application defines a particular pair of destination transport addresses (one network address plus a pair of ports for RTP and RTCP). In a multimedia session, each medium is carried in a separate RTP session, with its own RTCP packets reporting the reception quality for that session. For example, audio and video would travel on separate RTP sessions, enabling a receiver to select whether or not to receive a particular medium.

An audio-conferencing scenario presented in RFC 1889 illustrates the use of RTP. Suppose each participant sends audio data in segments of 20 ms duration. Each segment of audio data is preceded by an RTP header, and then the resulting RTP message is placed in a UDP packet. The RTP header indicates the type of audio encoding that is used, e.g., PCM. Users can opt to change the encoding during a conference in reaction to network congestion or, for example, to accommodate low-bandwidth requirements of a new conference participant. Timing information and a sequence number in the RTP header are used by the receivers to reconstruct the timing produced by the source, so that in this example, audio segments are contiguously played out at the receiver every 20 ms.

RTP fixed header fields

The RTP header has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: RTP Packet Header

The first twelve octets are present in every RTP packet, while the list of CSRC (contributing source) identifiers is present only when inserted by a mixer.The fields have the following meaning: **version (V): 2 bits**. Version of RTP. The newest version is 2.

padding (P): 1 bit. If set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored.

extension (X): 1 bit. If set, the fixed header is followed by exactly one header extension.

CSRC count (CC): 4 bits. The number of CSRC identifiers that follow the fixed header. This number is more than one if the payload of the RTP packet contains data from several sources.

marker (M): 1 bit. Defined by a profile, the marker is intended to allow significant events such as frame boundaries to be marked in the packet stream.

payload type (PT): 7 bits. Identifies the format of the RTP payload and determines its interpretation by the application.

sequence number: 16 bits. Increments by one for each RTP data packet sent, may be used by the receiver to detect packet loss and to restore packet sequence. The initial value is randomly set.

timestamp: 32 bits. The sampling instant of the first octet in the RTP data packet. May be used for synchronization and jitter calculations. The initial value is randomly set.

SSRC: 32 bits. Randomly chosen number to distinguish synchronization sources within the same RTP session. It indicates where the data was combined, or the source of the data if there is only one source.

CSRC list: 0 to 15 items, 32 bits each. Contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field.

RTCP---Real-Time Control Protocol

RTCP is the control protocol designed to work in conjunction with RTP. It is standardized in RFC 1889 and 1890. In an RTP session, participants periodically send RTCP packets to convey feedback on quality of data delivery and information of membership. RFC 1889 defines five RTCP packet types to carry control information. These five types are:

•**RR:** receiver report. Receiver reports are generated by participants that are not active senders. They contain reception quality feedback about data delivery, including the highest packets number received, the number of packets lost, inter-arrival jitter, and timestamps to calculate the round-trip delay between the sender and the receiver.

•**SR**: sender report. Sender reports are generated by active senders. In addition to the reception quality feedback as in RR, they contain a sender information section, providing information on inter-media synchronization, cumulative packet counters, and number of bytes sent.

•**SDES**: source description items. They contains information to describe the sources.

•**BYE**: indicates end of participation.

•**APP**: application specific functions. It is now intended for experimental use as new applications and new features are developed.

Through these control information packets, RTCP provides the following services:

•QoS monitoring and congestion control

This is the primary function of RTCP. RTCP provides feedback to an application about the quality of data distribution. The control information is useful to the senders, the receivers and third-party monitors. The sender can adjust its transmission based on the receiver report feedback. The receivers can determine whether a congestion is local, regional or global. Network managers can evaluate the network performance for multicast distribution.

•source identification

In RTP data packets, sources are identified by randomly generated 32-bit identifiers. These identifiers are not convenient for human users. RTCP SDES (source description) packets contain textual information called *canonical names* as globally unique identifiers of the session participants. It may include user's name, telephone number, email address and other information.

•inter-media synchronization

RTCP sender reports contain an indication of real time and the corresponding RTP timestamp. This can be used in inter-media synchronization like lip synchronization in video.

•control information scaling

RTCP packets are sent periodically among participants. When the number of participants increases, it is necessary to balance between getting up-to-date control information and limiting the control traffic. In order to scale up to large multicast groups, RTCP has to prevent the control traffic from overwhelming network resources. RTP limits the control traffic to at most 5% of the overall session traffic. This is enforced by adjusting the RTCP generating rate according to the number of participants.

RTP features

•RTP provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. But RTP itself does not provide any mechanism to ensure timely delivery. It needs support from lower layers that actually have control over resources in switches and routers. RTP depends on RSVP to reserve resources and to provide the requested quality of service.

•RTP doesn't assume anything about the underlying network, except that it provides framing. RTP is typically run on the top of UDP to make use of its multiplexing and checksum service, but efforts have been made to make RTP compatible with other transport protocols, such as ATM AAL5 and IPv6.

•Unlike usual data transmission, RTP does not offer any form of reliability or flow/congestion control. It provides timestamps, sequence numbers as hooks for adding reliability and flow/congestion control, but how to implement is totally left to the application.

•RTP is a protocol framework that is deliberately not complete. It is open to new payload formats and new multimedia software. By adding new profile and payload format specifications, one can tailor RTP to new data formats and new applications.

•RTP/RTCP provides functionality and control mechanisms necessary for carrying real-time content. But RTP/RTCP itself is not responsible for the higher-level tasks like assembly and synchronization. These have to be done at application level.

•The flow and congestion control information of RTP is provided by RTCP sender and receiver reports.

RTP Implementation resources

RTP is an open protocol that does not provide pre-implemented system calls. Implementation is tightly coupled to the application itself. Application developers have to add the complete functionality in the application layer by themselves. However, it is always more more efficient to share and reuse code rather than starting from scratch. The RFC 1889 specification itself contains numerous code segments that can be borrowed directly to the applications. There are some implementations with source code available on the web for evaluation and educational purposes. Many modules in the source code can be usable with minor modifications. The following is a list of useful resources:

•**vat** (http://www-nrg.ee.lbl.gov/vat/)
•**rtptools**(ftp://ftp.cs.columbia.edu/pub/schulzrinne/rtptools/)
•**NeVoT**(http://www.cs.columbia.edu/~hgs/rtp/nevot.html)
•**RTP page** (http://www/cs.columbia.edu/~hgs/rtp) maintained by Henning Schulzrinne is a very complete reference.
•**RTP Library** (http://www.cs.columbia.edu/~jdrosen/rtp_api.html) provides a high level interface for developing applications that make use of the Real Time Transport Protocol (RTP).

RTSP---Real-Time Streaming Protocol

Instead of storing large multimedia files and playing back, multimedia data is usually sent across the network in streams. Streaming breaks data into packets with size suitable for transmission between the servers and clients. The real-time data flows through the transmission, decompressing and playing back pipeline just like a water stream. A client can play the first packet, decompress the second, while receiving the third. Thus the user can start enjoying the multimedia without waiting to the end of transmission.

RTSP, the Real Time Streaming Protocol, is a client-server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over IP network. It provides "VCR-style" remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. Sources of data include both live data feeds and stored clips.

RTSP is an application-level protocol designed to work with lower-level protocols like RTP, RSVP to provide a complete streaming service over internet. It provides means for choosing delivery channels (such as UDP, multicast UDP and TCP), and delivery mechanisms based upon RTP. It works for large audience multicast as well as single-viewer unicast.

Development

RTSP was jointly developed by RealNetworks, Netscape Communications and Columbia University. It was developed from the streaming practice and experience of RealNetworks' RealAudio and Netscape's LiveMedia. The first draft of RTSP protocol was submitted to IETF on October 9, 1996 for consideration as an Internet Standard. Since then, it has gone through significant changes. The latest version is draft-ietf-mmusic-rtsp-07.

Although RTSP is still an internet draft at IETF and is likely to have significant changes, products using RTSP are available today. The major online players, Netscape, Apple, IBM, Silicon Graphics, VXtreme, Sun and other companies, had claimed their support to RTSP, though Microsoft's absence is conspicuous.

RTSP operations and methods

RTSP establishes and controls streams of continuous audio and video media between the media servers and the clients. A *media server* provides playback or recording services for the media streams while a *client* requests continuous media data from the media server. RTSP is the "network remote control" between the server and the client. It provides the following operations:

- **Retrieval of media from media server**: The client can request a presentation description, and ask the server to setup a session to send the requested data.

- **Invitation of a media server to a conference**: The media server can be invited to the conference to play back media or to record a presentation.

- **Adding media to an existing presentation**: The server or the client can notify each other about any additional media becoming available.

RTSP aims to provide the same services on streamed audio and video just as HTTP does for text and graphics. It is designed intentionally to have similar syntax and operations so that most extension mechanisms to HTTP can be added to RTSP.

In RTSP, each presentation and media stream is identified by an RTSP URL. The overall presentation and the properties of the media are defined in a presentation description file, which may include the encoding, language, RTSP URLs, destination address, port, and other parameters. The presentation description file can be obtained by the client using HTTP, email or other means.

But RTSP differs from HTTP in several aspects. First, while HTTP is a stateless protocol, an RTSP server has to maintain "session states" in order to correlate RTSP requests with a stream. Second, HTTP is basically an asymmetric protocol where the client issues requests and the server responds, but in RTSP both the media server and the client can issue requests. For example the server can issue an request to set playing back parameters of a stream.

In the current version, the services and operations are supported through the following methods:

•**OPTIONS:** The client or the server tells the other party the options it can accept.

•**DESCRIBE**: The client retrieves the description of a presentation or media object identified by the request URL from the server.

•**ANNOUNCE**: When sent from client to server, ANNOUNCE posts the description of a presentation or media object identified by the request URL to a server. When sent from server to client, ANNOUNCE updates the session description in real-time.

•**SETUP**: The client asks the server to allocate resources for a stream and start an RTSP session.

•**PLAY**: The client asks the server to start sending data on a stream allocated via SETUP.

•**PAUSE**: The client temporarily halts the stream delivery without freeing server resources.

•**TEARDOWN**: The client asks the server to stop delivery of the specified stream and free the resources associated with it.

•**GET_PARAMETER**: Retrieves the value of a parameter of a presentation or a stream specified in the URI.

•**SET_PARAMETER**: Sets the value of a parameter for a presentation or stream specified by the URI.

•**REDIRECT**: The server informs the clients that it must connect to another server location. The mandatory location header indicates the URL the client should connect to.

•**RECORD**: The client initiates recording a range of media data according to the presentation description.

Note that some of these methods can be sent either from the server to the client or from the client to the server, but others can only be sent in one direction. Not all these methods are necessary in a fully functional server. For example, a media server with live feeds may not support the PAUSE method.

RTSP requests are usually sent on a channel independent of the data channel. They can be transmitted in persistent transport connections, or as a one connection per request/response transaction, or in connectionless mode.

RTSP features

•RTSP is an application level protocol with syntax and operations similar to HTTP, but works for audio and video. It uses URLs like those in HTTP.
•An RTSP server needs to maintain states, using SETUP, TEARDOWN and other methods.
•RTSP messages are be carried out-of-band. The protocol for RTSP may be different from the data delivery protocol.
•Unlike HTTP, in RTSP both servers and clients can issue requests.
•RTSP is implemented on multiple operating system platforms, it allows interoperability between clients and servers from different manufacturers.

RTSP implementation resources

Although RTSP is still an IETF draft, there are a few implementations already available on the web. The following is a collection of useful implementation resources:

RTSP Reference Implementation(http://www.real.com/devzone/library/fireprot/rtsp/reference.html) This is a source code testbed for the standards community to experiment with RTSP compatibility.

RealMedia SDK(http://www.realnetworks.com/devzone/tools/index.html) This is an open, cross platform, client-server system that implementors can create RTSP-based streaming applications. It includes a working RTSP client and server, as well as the components to quickly create RTSP-based applications which stream arbitrary data types and file formats.

W3C's Jigsaw(http://www.w3.org/Jigsaw/) A Java-based web server. The RTSP server in the latest beta version was written in Java.

IBM's RTSP Toolkit(http://www.research.ibm.com/rtsptoolkit/) IBM's toolkits derived from from tools developed for ATM/video research and other applications in 1995-1996. Its shell-based implementation illustrates usefulness of the RTSP protocol for non-multimedia applications.

Session Announcement Protocol (SAP)

The Session Announcement Protocol (SAP) must be one of the simplest protocols around. To announce a multicast session, the session creator merely multicasts packets periodically to a well-known multicast group carrying an SDP description of the session that is going to take place. People that wish to know which sessions are going to be active simply listen to the same well-known multicast group, and receive those announcement packets. Of course, the protocol gets a little more complex when we take security and caching into account, but basically that's it.

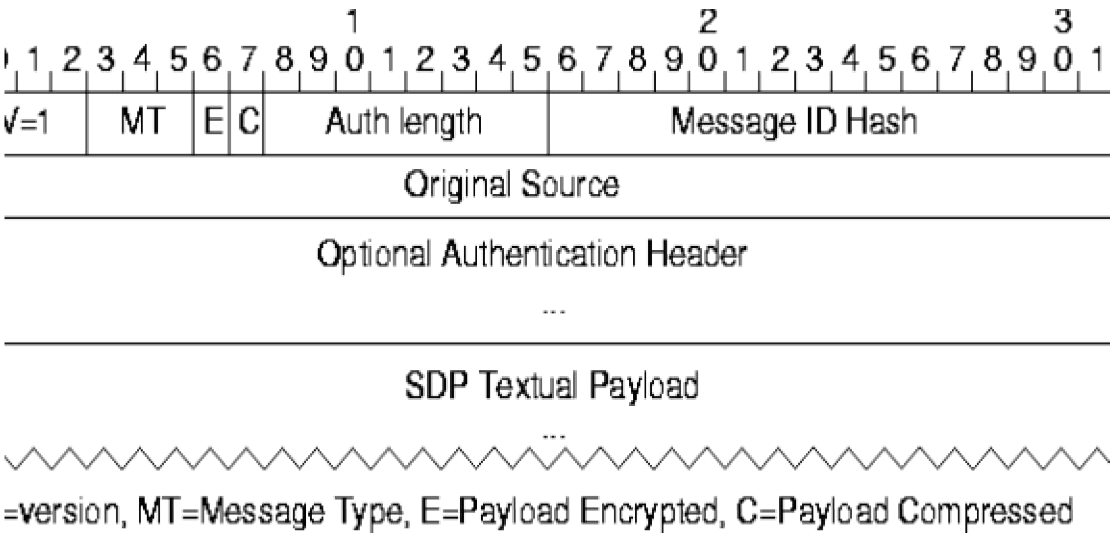


Figure 5: Session Announcement Packets

The SAP packet format (for IPv4) is shown in figure 5. The Message Type (MT) field indicates whether this packet announces a session, or deletes an announcement. One bit (E) indicates whether or not the payload is encrypted and one bit (C) indicates whether or not the payload is compressed. The combination of *message ID hash* and *original source* is supposed to provide a unique announcement ID that can be used to identify this particular version of this particular session. This is useful for caching or for ignoring packets that we previously failed to decrypt, but as this announcement ID is not guaranteed to be unique, care must be taken to also check the packet length and periodically check the packet contents themselves.

SAP announcements can be authenticated by including a digital signature of the payload in the optional *authentication header*. Both PGP and PKCS#7 based digital signatures are currently supported in the SAP protocol, although currently not many announcements are authenticated. As multicast-based sessions become more popular and people attempt to subvert them, this will no-doubt change.

SAP announcements can also be encrypted. However, this does not mean that the standard way to have small private wide-area conferences will be to announce them with encrypted SAP - the Session Initiation Protocol is a more appropriate mechanism for such conferences. The main uses for encrypted SAP announcements would appear to be in intranet environments where the SAP announcement bother few additional people, or for very large sessions where members are charged to participate. In the latter case, it would probably be a good idea to provide an additional level of access control beyond SAP encryption because it is easy for one misbehaving participant to leak a SAP key to other potential participant unless the keys are embedded in hardware as they are with satellite television smart cards.

The announcement rate for SAP is quite low, with typically several minutes between repeated announcements of the same session. Thus a user starting a SAP receiver will have to wait for a few minutes before seeing all the sessions. This is normally solved by caching - either by running a SAP receiver in the background all the time to keep the cache up-to-date, or by going to a local SAP proxy on startup and requesting a cache download.

It should be clear that although SAP will scale to any number of receivers, it will not scale to huge numbers of sessions. It is currently an IETF Experimental Standard, which reflects this belief that we will eventually need to replace it with something different. In the meantime though, it is a great way to bootstrap the use of IP multicast, and with appropriate caching, SAP will be OK with up to a few thousand sessions advertised.

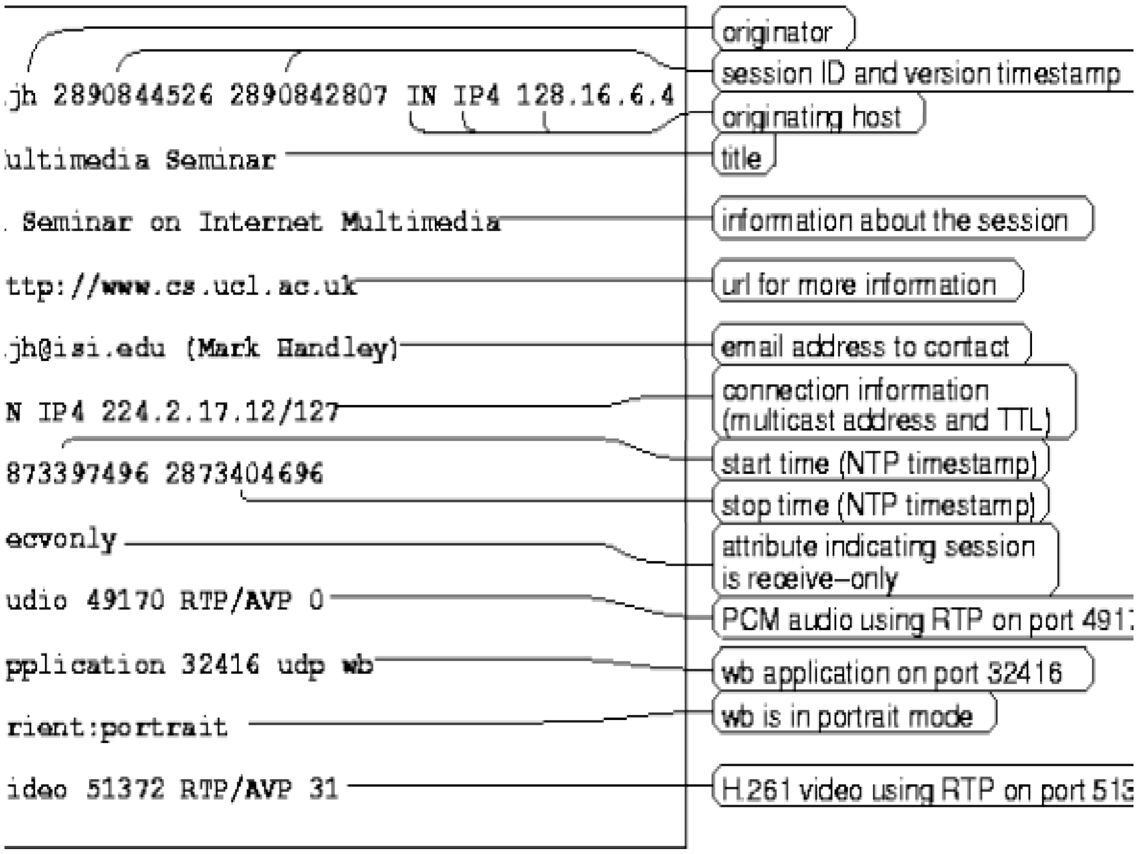
SAP should be used for sessions of some public interest where the participants are not known in advance. If you know who you want in your session, a better mechanism is to explicitly invite them using SIP.

Session Description Protocol (SDP)

A session description expressed in SDP is a short structured textual description of the name and purpose of the session, and the media, protocols, codec formats, timing and transport information that are required to decide whether a session is likely to be of interest and to know how to start media tools to participate in the session.

SDP is purely a format for session description - it does not incorporate a transport protocol, and is intended to use different transport protocols as appropriate, including the Session Announcement Protocol, Session Initiation Protocol, Real-Time Streaming Protocol, electronic mail using the MIME extensions, and the Hyper-Text Transport Protocol (HTTP).

Although SDP is not intended to be read by humans, it is relatively easily understood. An example session description is shown in figure 6



```
.jh 2890844526 2890842807 IN IP4 128.16.6.4
ultimedia Seminar
. Seminar on Internet Multimedia
ttp://www.cs.ucl.ac.uk
.jh@isi.edu (Mark Handley)
N IP4 224.2.17.12/127
873397496 2873404696
ecvonly
udio 49170 RTP/AVP 0
pplication 32416 udp wb
rient:portrait
ideo 51372 RTP/AVP 31
```

originator
session ID and version timestamp
originating host
title
information about the session
url for more information
email address to contact
connection information (multicast address and TTL)
start time (NTP timestamp)
stop time (NTP timestamp)
attribute indicating session is receive-only
PCM audio using RTP on port 491
wb application on port 32416
wb is in portrait mode
H.261 video using RTP on port 51

Figure 6: Annotated SDP Session Description

Even without the annotation, it would be easy to guess that this session is entitled ``Multimedia Seminar'', that it will use audio, video and an application called ``wb'', and that someone called ``Mark Handley'' is responsible for the session.

However this SDP also indicates the precise timing of the session (the ``t=" line gives start and stop times), that the session is multicast to group 224.2.17.12 with a TTL of 127, the audio is 8KHz μ law carried by RTP to UDP port 49170, the video is H.261 encoded, also over RTP but to port 51372, and the whiteboard program should be started up in portrait mode using port 32416.

Thus SDP includes the *session name* and a description of its *purpose*, the *times* the session is active, the *media* comprising the session, and information to receive those media (addresses, ports, formats and so on).

As resources necessary to participate in a session may be limited, some additional information may also be desirable, including information about the bandwidth to be used by the conference and contact information for the person responsible for the session.

In general, SDP must convey sufficient information to be able to join a session (with the possible exception of encryption keys) and to announce the resources to be used to non-participants that may need to know.

Timing Information

Sessions may either be bounded or unbounded in time. Whether or not they are bounded, they may be only active at specific times. SDP can convey an arbitrary list of start and stop times bounding the session. For each bound, repeat times such as ``every Wednesday at 10am for one hour" can be specified. This timing information is globally consistent, irrespective of local time zone or daylight saving time because it is given in UTC. Modifiers can be specified to uniquely apply offsets if the session crosses a change to or from daylight saving time.

SDP: Summary

SDP originated for announcing multicast sessions, and it serves that task well. It was not originally intended to deal with unicast sessions, but it works well enough in the context of SIP and RTSP. It was certainly never intended to be a content *negotiation* mechanism, because it lacks any clean way to group different options together into capability sets. Although it can serve this role, when it does so, the limitations of its original design purpose start to show through somewhat.

SDP is in widespread use in SAP, SIP, H.332, and RTSP, and has even been proposed for use in the context of advanced television.

Overview of the Session Initiation Protocol

This chapter provides an overview of SIP. It includes the following sections:

- Introduction to SIP
- Components of SIP
- How SIP Works
- SIP Versus H.323

Introduction to SIP

Session Initiation Protocol (SIP) is the Internet Engineering Task Force's (IETF's) standard for multimedia conferencing over IP. SIP is an ASCII-based, application-layer control protocol (defined in RFC 2543) that can be used to establish, maintain, and terminate calls between two or more end points.

Like other VoIP protocols, SIP is designed to address the functions of signaling and session management within a packet telephony network. *Signaling* allows call information to be carried across network boundaries. *Session management* provides the ability to control the attributes of an end-to-end call.

SIP provides the capabilities to:

- Determine the location of the target end point—SIP supports address resolution, name mapping, and call redirection.

- Determine the media capabilities of the target end point—Via Session Description Protocol (SDP), SIP determines the "lowest level" of common services between the end points. Conferences are established using only the media capabilities that can be supported by all end points.

- Determine the availability of the target end point—If a call cannot be completed because the target end point is unavailable, SIP determines whether the called party is already on the phone or did not answer in the allotted number of rings. It then returns a message indicating why the target end point was unavailable.

- Establish a session between the originating and target end point—If the call can be completed, SIP establishes a session between the end points. SIP also supports midcall changes, such as the addition of another end point to the conference or the changing of a media characteristic or codec.

- Handle the transfer and termination of calls—SIP supports the transfer of calls from one end point to another. During a call transfer, SIP simply establishes a session between the transferee and a new end point (specified by the transferring party) and terminates the session between the transferee and the transferring party. At the end of a call, SIP terminates the sessions between all parties.

Conferences can consist of two or more users and can be established using multicast or multiple unicast sessions.

Note The term *conference* means an established session (or *call*) between two or more end points. In this document, the terms conference and call are used interchangeably.

Components of SIP

SIP is a peer-to-peer protocol. The peers in a session are called User Agents (UAs). A user agent can function in one of the following roles:

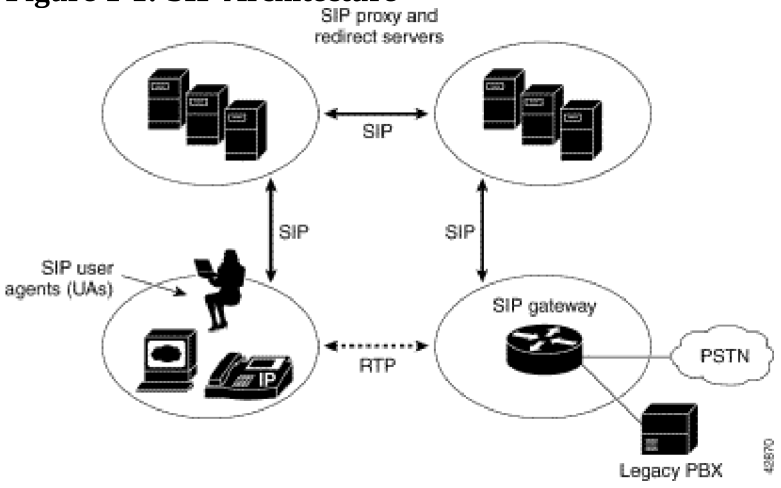
- User agent client (UAC)—A client application that initiates the SIP request.
- User agent server (UAS)—A server application that contacts the user when a SIP request is received and that returns a response on behalf of the user.

Typically, a SIP end point is capable of functioning as both a UAC and a UAS, but functions only as one or the other per transaction. Whether the endpoint functions as a UAC or a UAS depends on the UA that initiated the request.

From an architecture standpoint, the physical components of a SIP network can be grouped into two categories: clients and servers. Figure 1-1 illustrates the architecture of a SIP network.

Note In addition, the SIP servers can interact with other application services, such as Lightweight Directory Access Protocol (LDAP) servers, location servers, a database application, or an extensible markup language (XML) application. These application services provide back-end services such as directory, authentication, and billing services.

Figure 1-1: SIP Architecture



SIP Clients

SIP clients include:

- Phones—Can act as either a UAS or UAC. Softphones (PCs that have phone capabilities installed) and Cisco SIP IP phones can initiate SIP requests and respond to requests.

- Gateways—Provide call control. Gateways provide many services, the most common being a translation function between SIP conferencing endpoints and other terminal types. This function includes translation between transmission formats and between communications procedures. In addition, the gateway translates between audio and video codecs and performs call setup and clearing on both the LAN side and the switched-circuit network side.

SIP Servers

SIP servers include:

- Proxy server—The proxy server is an intermediate device that receives SIP requests from a client and then forwards the requests on the client's behalf. Basically, proxy servers receive SIP messages and forward them to the next SIP server in the network. Proxy servers can provide functions such as authentication, authorization, network access control, routing, reliable request retransmission, and security.

- Redirect server—Provides the client with information about the next hop or hops that a message should take and then the client contacts the next hop server or UAS directly.

- Registrar server—Processes requests from UACs for registration of their current location. Registrar servers are often co-located with a redirect or proxy server.

How SIP Works

SIP is a simple, ASCII-based protocol that uses requests and responses to establish communication among the various components in the network and to ultimately establish a conference between two or more end points.

Users in a SIP network are identified by unique SIP addresses. A SIP address is similar to an e-mail address and is in the format of sip:*userID@gateway*.com. The user ID can be either a user name or an E.164 address.

Users register with a registrar server using their assigned SIP addresses. The registrar server provides this information to the location server upon request.

When a user initiates a call, a SIP request is sent to a SIP server (either a proxy or a redirect server). The request includes the address of the caller (in the From header field) and the address of the intended callee (in the To header field). The following sections provide simple examples of successful, point-to-point calls established using a proxy and a redirect server.

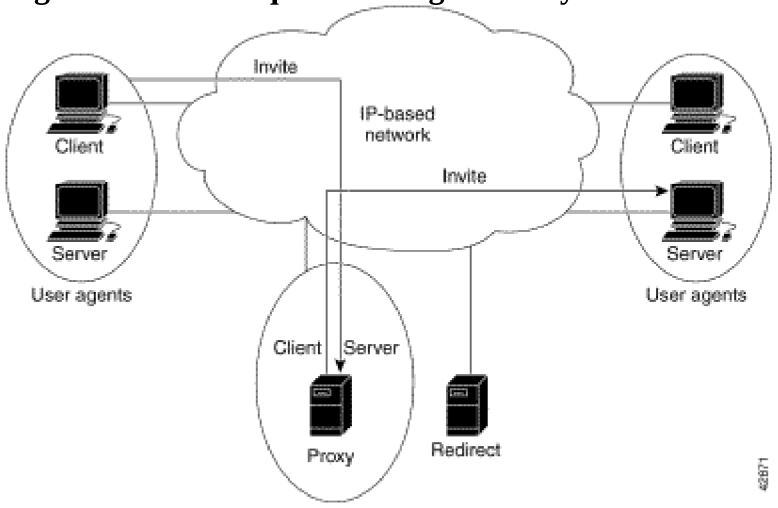
Over time, a SIP end user might move between end systems. The location of the end user can be dynamically registered with the SIP server. The location server can use one or more protocols (including finger, rwhois, and LDAP) to locate the end user. Because the end user can be logged in at more than one station and because the location server can sometimes have inaccurate information, it might return more than one address for the end user. If the request is coming through a SIP proxy server, the proxy server will try each of the returned addresses until it locates the end user. If the request is coming through a SIP redirect server, the redirect server forwards all the addresses to the caller in the Contact header field of the invitation response.

For more information, see RFC 2543—*SIP: Session Initiation Protocol*, which can be found at http://www.faqs.org/rfcs/.

Using A Proxy Server

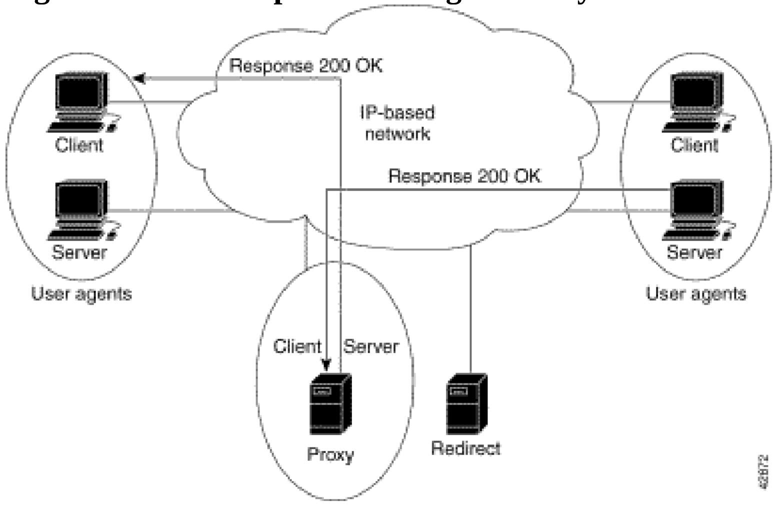
If a proxy server is used, the caller UA sends an INVITE request to the proxy server, the proxy server determines the path, and then forwards the request to the callee (as shown in Figure 1-2).

Figure 1-2: SIP Request Through a Proxy Server



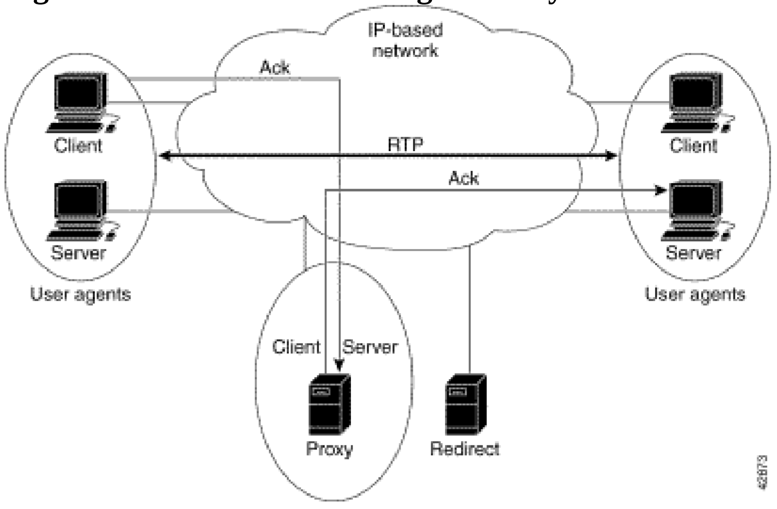
The callee responds to the proxy server, which in turn, forwards the response to the caller (as shown in Figure 1-3).

Figure 1-3: SIP Response Through a Proxy Server



The proxy server forwards the acknowledgments of both parties. A session is then established between the caller and callee. Real-time Transfer Protocol (RTP) is used for the communication between the caller and the callee (as shown in Figure 1-4).

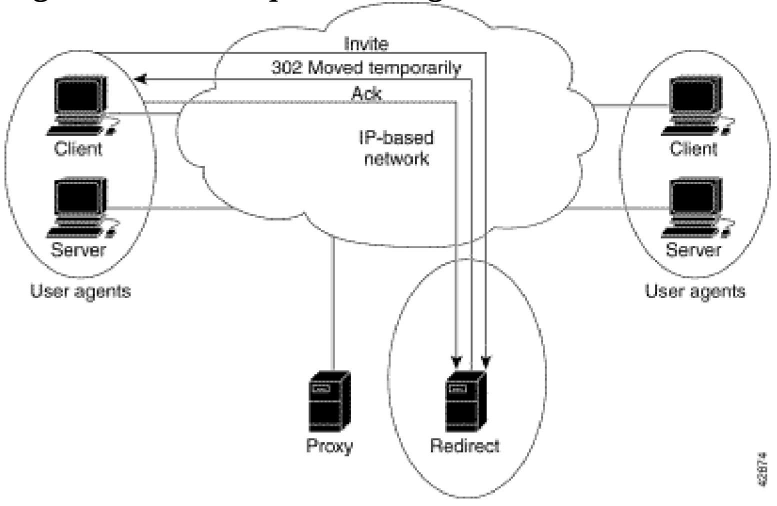
Figure 1-4: SIP Session Through a Proxy Server



Using a Redirect Server

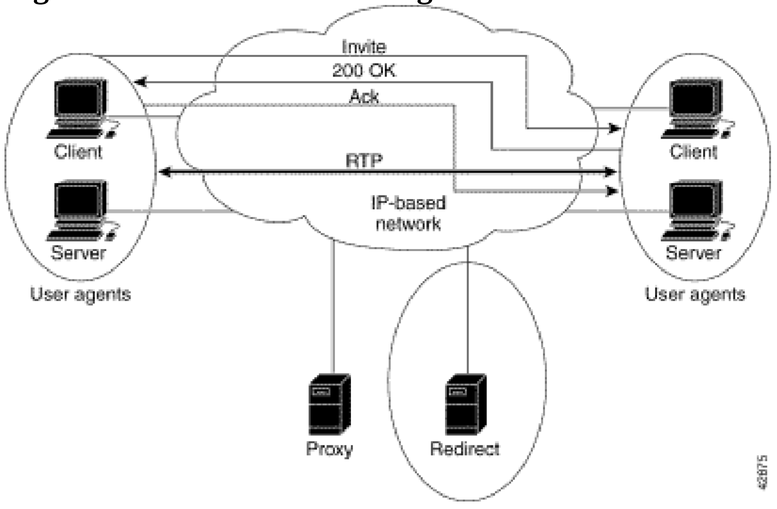
If a redirect server is used, the caller UA sends an INVITE request to the redirect server, the redirect server contacts the location server to determine the path to the callee, and then the redirect server sends that information back to the caller. The caller then acknowledges receipt of the information (as shown in Figure 1-5).

Figure 1-5: SIP Request Through a Redirect Server



The caller then sends a request to the device indicated in the redirection information (which could be the callee or another server that will forward the request). Once the request reaches the callee, it sends back a response and the caller acknowledges the response. RTP is used for the communication between the caller and the callee (as shown in Figure 1-6).

Figure 1-6: SIP Session Through a Redirect Server



SIP Versus H.323

In addition to SIP, there are other protocols that facilitate voice transmission over IP. One such protocol is H.323. H.323 originated as an International Telecommunications Union (ITU) multimedia standard and is used for both packet telephony and video streaming. The H.323 standard incorporates multiple protocols, including Q.931 for signaling, H.245 for negotiation, and Registration Admission and Status (RAS) for session control. H.323 was the first standard for call control for VoIP and is supported on all Cisco Systems' voice gateways.

SIP and H.323 were designed to address session control and signaling functions in a distributed call control architecture. Although SIP and H.323 can also be used to communicate to limited intelligence end points, they are especially well-suited for communication with intelligent end points.

Table 1-1 provides a brief comparison of SIP and H.323.

Table 1-1: SIP versus H.323

Aspect	**SIP**	**H.323**
Clients	Intelligent	Intelligent
Network intelligence and services	Provided by servers (Proxy, Redirect, Registrar)	Provided by gatekeepers
Model used	Internet/WWW	Telephony/Q.SIG
Signaling protocol	UDP or TCP	TCP (UDP is optional in Version 3)
Media protocol	RTP	RTP
Code basis	ASCII	Binary (ASN.1 encoding)
Other protocols used	IETF/IP protocols, such as SDP, HTTP/1.1, IPmc, and MIME	ITU / ISDN protocols, such as H.225, H.245, and H.450
Vendor interoperability	Widespread	Widespread

Although SIP messages are not directly compatible with H.323, both protocols can coexist in the same packet telephony network if a device that supports the interoperability is available.

For example, a call agent could use H.323 to communicate with gateways and use SIP for inter-call agent signaling. Then, after the bearer connection is set up, the bearer information flows between the different gateways as an RTP stream.

ΒΙΒΛΙΟΓΡΑΦΙΑ - BIBLIOGRAPHY

http://www.research.ibm.com/rtsptoolkit/

http://demo.freshwater.com/SiteScope/docs/RTSPMonitor.htm

http://vod.sourceforge.net/index.shtml

http://live.sourceforge.net/

http://developer.apple.com/darwin/projects/streaming/

http://www.icsd.aegean.gr/lecturers/arouskas/courses/mps_internet/schedule_lectures.htm

http://www.medialab.ece.ntua.gr/medialab/education/education.html

http://www.medialab.ntua.gr/multinew/extra/append6.htm

http://www.cisco.com/univercd/cc/td/doc/product/voice/sipsols/biggulp/bgsipov.htm

http://smuhandouts.com/F8313/

http://www4.cs.fau.de/Projects/JRTP/pmt/node25.html#454

ftp://catarina.usc.edu/pub/mitzel/Infocom94/infocom94.ps

ftp://parcftp.xerox.com/pub/net-research/rsvp.ps.Z

http://network-services.uoregon.edu/~ursula/diplom_lt.ps

http://www.cis.ohio-state.edu/htbin/rfc/rfc1889.html

www.cis.ohio-state.edu/htbin/rfc/rfc1890.html

http://http.cs.berkeley.edu/~amit/research/Postscript/estab-nossdav.ps

http://http.cs.berkeley.edu/~amit/research/Postscript/adv-nossdav.ps,

http://http.cs.berkeley.edu/~amit/research/Postscript/infocom95.ps, Proc.

http://www.cs.columbia.edu/~hgs/rtp/

http://www.ipmulticast.com/community/whitepapers/highprot.html.

ftp://ftp.isi.edu/in-notes/rfc2035.txt

ftp://ftp.isi.edu/in-notes/rfc2032.txt

ftp://ftp.isi.edu/in-notes/rfc2038.txt

ftp://ftp.isi.edu/in-notes/rfc2029.txt

ftp://ftp.ietf.org/internet-drafts/draft-ietf-avt-profile-new-01.txt

http://www.cs.columbia.edu/~hgs/rtsp/draft/draft-ietf-mmusic-stream-00.txt

http://www.realnetworks.com/devzone/library/rtsp/index.html

http://www.cs.columbia.edu/~hgs/rtsp/